**eHeritage.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

using ESRI.ArcGISExplorer;

using ESRI.ArcGISExplorer.Application;

using ESRI.ArcGISExplorer.Mapping;

using ESRI.ArcGISExplorer.Geometry;

using ESRI.ArcGISExplorer.Data;

using ESRI.ArcGISExplorer.Threading;

/\*

eHeritage Dunedin - ESRI Extension for ArcGIS Explorer.

Author: Adon Moskal

March 2010 - June 2010

\*/

namespace ArcGISExplorerExtension

{

public class eHeritage : ESRI.ArcGISExplorer.Application.Extension

{

MapDisplay md = ESRI.ArcGISExplorer.Application.Application.ActiveMapDisplay;

public override void OnStartup()

{

//Creates a new form associated with this extension on startup

Form1 form = new Form1();

//sets the form width to the size of the screen

form.Width = Screen.PrimaryScreen.Bounds.Width;

form.Height = 200;

//sets form location to bottom of the screen

System.Drawing.Point myPt = new System.Drawing.Point(0, Screen.PrimaryScreen.Bounds.Height - 200);

form.Location = myPt;

form.Show();

form.Activate();

}

public override void OnShutdown()

{

}

}

}

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

using ESRI.ArcGISExplorer;

using ESRI.ArcGISExplorer.Application;

using ESRI.ArcGISExplorer.Mapping;

using ESRI.ArcGISExplorer.Geometry;

using ESRI.ArcGISExplorer.Data;

using ESRI.ArcGISExplorer.Threading;

using System.IO;

namespace ArcGISExplorerExtension

{

public partial class Form1 : Form

{

//data members

MapDisplay md = ESRI.ArcGISExplorer.Application.Application.ActiveMapDisplay;

TYear currYear;

TEngine currEngine;

ESRI.ArcGISExplorer.Mapping.View currView;

DataView dv;

ListBox newBox;

Graphics displayGraphics;

double bearing;

List<int> yearList;

Image compass;

ImageList yearNumbers;

Form picForm;

string path;

public const int STARTYEAR = 1846;

public const int ENDYEAR = 2009;

public const int COMPASS\_POS = 350;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

displayGraphics = this.CreateGraphics();

currEngine = new TEngine(md);

path = currEngine.getPath();

currView = currEngine.getView();

yearList = new List<int>();

compass = Bitmap.FromFile(path + "Images\\compass1.png");

yearNumbers = new ImageList();

string[] numberFiles = Directory.GetFiles(path + "Images\\", "\*.jpg");

foreach (string nfile in numberFiles)

{

Image newImage = Image.FromFile(nfile);

yearNumbers.Images.Add(newImage);

}

//check to see if a model exists for a year and if so, add it to the list

for (int i = STARTYEAR; i < ENDYEAR; i++)

{

if (File.Exists(path + "Output\\" + i.ToString() + "\\" + i.ToString() + ".kmz"))

{

yearList.Add(i);

}

}

md.GraphicClicked += new EventHandler<GraphicMouseEventArgs>(md\_GraphicClicked);

md.MapItemClicked += new EventHandler<MapItemMouseEventArgs>(md\_MapItemClicked);

//dynamically create the number of notches for the year trackbar

trackBar1.Maximum = yearList.Count-1;

trackBar1.Value = 0;

startLabel.Text = yearList[0].ToString();

EndLabel.Text = yearList[yearList.Count-1].ToString();

newBox = new ListBox();

bearing = new double();

timer1.Enabled = true;

//make a new form to display pictures in

picForm = new Form();

picForm.Width = md.Size.Width / 2;

picForm.Height = md.Size.Height / 2;

System.Drawing.Point myPt = new System.Drawing.Point(md.Size.Width / 2 - picForm.Width / 2, md.Size.Height / 2 - picForm.Height / 2);

picForm.Location = myPt;

picForm.TopMost = true;

picForm.FormClosing += new FormClosingEventHandler(picForm\_FormClosing);

}

void picForm\_FormClosing(object sender, FormClosingEventArgs e)

{

//stop the form from closing and simply hide it from view

e.Cancel = true;

picForm.Hide();

}

void md\_GraphicClicked(object sender, GraphicMouseEventArgs e)

{

//clear any image overlays on the screen

for (int i=0;i<md.Map.ChildItems.Count;i++)

{

if (md.Map.ChildItems[i] is ImageOverlay)

md.Map.ChildItems.Remove(md.Map.ChildItems[i]);

continue;

}

//filter the shapefile data by the name of the clicked label

dv = currYear.MainDV;

dv.RowFilter = "Name = '" + e.Graphic.Label + "'";

//if any data is returned, add it to the listbox

if (dv.Count != 0)

{

DataRow R = dv.ToTable().Rows[0];

listBox1.Items.Clear();

listBox1.Items.Add(R[1].ToString());

listBox1.Items.Add("Built in: " + R[2].ToString());

}

//create list of available images

List<PictureBox> pictureList = new List<PictureBox>();

int lastWidth = 0;

//remove existing image thumbnails from the screen

for (int i = this.Controls.Count - 1; i >= 0; i--)

{

PictureBox control = this.Controls[i] as PictureBox;

if (control == null)

continue;

control.Dispose();

}

//get all images associated with this building

string[] buildingFiles = Directory.GetFiles(path + "Output\\" + currYear.CurrYear, e.Graphic.Label + "\*.jpg");

for (int i=0;i<buildingFiles.Length;i++)

{

PictureBox pictureBox = new PictureBox();

//create a new image from the imagelist

Image photoImg = Image.FromFile(buildingFiles[i]);

//create the thumbnail for the image

Image thumbPhoto = photoImg.GetThumbnailImage(50, 50, null, new System.IntPtr());

pictureBox.Image = thumbPhoto;

pictureBox.Width = thumbPhoto.Width;

pictureBox.Height = thumbPhoto.Height;

pictureBox.Name = buildingFiles[i];

//make the thumbnail clickable

pictureBox.Click += new EventHandler(pictureBox\_Click);

//place the thumnnail on the form

this.Controls.Add(pictureBox);

pictureBox.Location = new System.Drawing.Point((listBox1.Location.X + listBox1.Width + 50) + i \* lastWidth, 50);

lastWidth = pictureBox.Width;

}

//refresh screen

System.Windows.Forms.Application.DoEvents();

}

void pictureBox\_Click(object sender, EventArgs e)

{

//remove any image overlays currently onscreen

for (int i = 0; i < md.Map.ChildItems.Count; i++)

{

if (md.Map.ChildItems[i] is ImageOverlay)

md.Map.ChildItems.Remove(md.Map.ChildItems[i]);

continue;

}

//show the picture form

picForm.Show();

picForm.Activate();

//remove existing images from the form

foreach (Control c in picForm.Controls)

{

picForm.Controls.Remove(c);

}

//display the clicked thumbnail as a full size image in the picture form

PictureBox pb = (PictureBox)sender;

PictureBox picBox = new PictureBox();

picBox.Width = picForm.Width;

picBox.Height = picForm.Height;

Image im = Image.FromFile(pb.Name);

picBox.SizeMode = PictureBoxSizeMode.StretchImage;

picBox.Load(pb.Name);

picForm.Controls.Add(picBox);

}

void md\_MapItemClicked(object sender, MapItemMouseEventArgs e)

{

//clear any image overlays on the screen

for (int i = 0; i < md.Map.ChildItems.Count; i++)

{

if (md.Map.ChildItems[i] is ImageOverlay)

md.Map.ChildItems.Remove(md.Map.ChildItems[i]);

continue;

}

//hide the default popup associated with kml models

md.HidePopups(true);

}

private void timer1\_Tick(object sender, EventArgs e)

{

//formulas for the compass

bearing = md.Rotation \* (Math.PI / 180);

double x = new double();

double y = new double();

x = 50 \* Math.Sin(bearing);

y = -50 \* Math.Cos(bearing);

//draw the compass to the form

displayGraphics.DrawImage(compass, COMPASS\_POS, 25);

displayGraphics.DrawLine(new Pen(new SolidBrush(Color.Red)), compass.Width / 2 + (COMPASS\_POS + 3), compass.Height / 2 + 24, (int)x + compass.Width / 2 + (COMPASS\_POS + 3), (int)y + compass.Height / 2 + 24);

//if the user if rotating the 3D world, refresh the screen

if (md.IsNavigating == true)

{

System.Windows.Forms.Application.DoEvents();

}

}

private void button1\_Click(object sender, EventArgs e)

{

//capture the year selected from the trackbar

string year = "";

year = yearList[trackBar1.Value].ToString();

//remove any image overlays currently on the screen

for (int i = 0; i < md.Map.ChildItems.Count; i++)

{

if (md.Map.ChildItems[i] is ImageOverlay)

md.Map.ChildItems.Remove(md.Map.ChildItems[i]);

continue;

}

//remove any image thumbnails currently on the screen

for (int i = this.Controls.Count - 1; i >= 0; i--)

{

PictureBox control = this.Controls[i] as PictureBox;

if (control == null)

continue;

control.Dispose();

}

//create a new current year

currYear = new TYear(md);

//show the current year data - model, map, shapefile data

currYear.showMap(path, year);

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

//update the number images as the trackbar is scrolled

drawYear();

}

public void drawYear()

{

//check which year the user has scrolled to

string yearTrack = yearList[trackBar1.Value].ToString();

yearNumbers.ImageSize = new Size(50,60);

//get the appropriate images for the year digits

for (int i = 0; i < 4; i++)

{

switch (yearTrack[i])

{

case '0': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 0);

break;

case '1': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 1);

break;

case '2': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 2);

break;

case '3': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 3);

break;

case '4': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 4);

break;

case '5': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 5);

break;

case '6': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 6);

break;

case '7': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 7);

break;

case '8': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 8);

break;

case '9': yearNumbers.Draw(displayGraphics, 20 + i \* (yearNumbers.ImageSize.Width + 5), 100, 9);

break;

}

}

System.Windows.Forms.Application.DoEvents();

}

private void viewBtn\_Click(object sender, EventArgs e)

{

//returns to the default view in case the user navigates away

md.ZoomTo(currView);

}

}

}

**TEngine.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

using ESRI.ArcGISExplorer;

using ESRI.ArcGISExplorer.Application;

using ESRI.ArcGISExplorer.Mapping;

using ESRI.ArcGISExplorer.Geometry;

using ESRI.ArcGISExplorer.Data;

using ESRI.ArcGISExplorer.Threading;

namespace ArcGISExplorerExtension

{

public class TEngine

{

//data members

string path;

string newpath;

protected MapDisplay md = ESRI.ArcGISExplorer.Application.Application.ActiveMapDisplay;

//constructor

public TEngine(MapDisplay md)

{

this.md = md;

}

//gets the working directory path of the eHeritage Database - used for loading data

public string getPath()

{

//get the user to select the working directory

MessageBox.Show("Please select a file to specify a working directory.");

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == DialogResult.OK)

{

path = System.IO.Path.GetDirectoryName(ofd.FileName);

}

MessageBox.Show("Working Directory: \"" + path + "\"", "Directory");

//format string so it can be used as a path

newpath = path.Replace("\\", "\\\\");

newpath = newpath + "\\\\";

return newpath;

}

//gets the default view on map load - used with the Default View button

public ESRI.ArcGISExplorer.Mapping.View getView()

{

ESRI.ArcGISExplorer.Mapping.View currView = new ESRI.ArcGISExplorer.Mapping.View("view1", md.GetViewpoint());

return currView;

}

}

}

**TYear.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

using ESRI.ArcGISExplorer;

using ESRI.ArcGISExplorer.Application;

using ESRI.ArcGISExplorer.Mapping;

using ESRI.ArcGISExplorer.Geometry;

using ESRI.ArcGISExplorer.Data;

using ESRI.ArcGISExplorer.Threading;

using System.Threading;

namespace ArcGISExplorerExtension

{

public class TYear

{

//data members

protected MapDisplay md = ESRI.ArcGISExplorer.Application.Application.ActiveMapDisplay;

protected RasterLayer ras;

protected KmlLayer kml;

public KmlLayer Kml

{

get { return kml; }

set { kml = value; }

}

protected FeatureLayer shape;

public FeatureLayer Shape

{

get { return shape; }

set { shape = value; }

}

protected DataView dv;

public DataView Dv

{

get { return dv; }

set { dv = value; }

}

protected DataTable dt;

protected DataTable mainDT;

protected DataView mainDV;

public DataView MainDV

{

get { return mainDV; }

set { mainDV = value; }

}

private string currYear;

public string CurrYear

{

get { return currYear; }

set { currYear = value; }

}

//constructor

public TYear(MapDisplay md)

{

this.md = md;

}

//methods

public void showMap(string newpath, string year)

{

currYear = year;

try

{

//connect to the Access DB

OleDbConnection con = new OleDbConnection(@"Provider=Microsoft.JET.OLEDB.4.0;"

+ @"data source=" + newpath + "eHeritage GeoDatabase.mdb");

con.Open();

DataSet ds = new DataSet();

DataSet mainDS = new DataSet();

OleDbDataAdapter adapter = new OleDbDataAdapter("Select \* from buildings\_" + year, con);

OleDbDataAdapter adapter2 = new OleDbDataAdapter("Select \* from T\_Building", con);

adapter.Fill(ds);

adapter2.Fill(mainDS);

con.Close();

dt = ds.Tables[0];

dv = dt.DefaultView;

mainDT = mainDS.Tables[0];

mainDV = mainDT.DefaultView;

//clear existing elements from the map display

md.Map.ChildItems.Clear();

md.Graphics.Clear();

//if a \*.png map exists for current year, load it into a new RasterLayer

if (System.IO.File.Exists(newpath + "Output\\" + year + "\\" + year + ".png"))

{

ras = RasterLayer.OpenRasterFile(newpath + "Output\\" + year + "\\" + year + ".png");

ras.BackgroundTransparentColor = Color.Black;

ras.Transparency = 20;

}

//open the kml model

kml = KmlLayer.Open(newpath + "Output\\" + year + "\\" + year + ".kmz");

//if a shapefile exists for current year, load it into a new FeatureLayer

if (System.IO.File.Exists(newpath + "Output\\" + year + "\\" + year + ".shp"))

shape = FeatureLayer.OpenShapefile(newpath + "Output\\" + year + "\\" + year + ".shp");

if (kml != null)

md.Map.ChildItems.Add(kml);

if (ras != null)

md.Map.ChildItems.Add(ras);

//for each "row" in the shapefile, add a label to the map in the centre of each building

if (shape != null)

{

foreach (Row R in shape.Table.GetRows())

{

ESRI.ArcGISExplorer.Geometry.Point p = R.Geometry.GetCenter();

Symbol s = Symbol.Marker.Transparent;

Graphic g = new Graphic(p, s);

g.Label = R.Values[4].ToString();

md.Graphics.Add(g);

}

}

}

catch (ConnectionException)

{

MessageBox.Show("Connection Error");

}

}

}

}